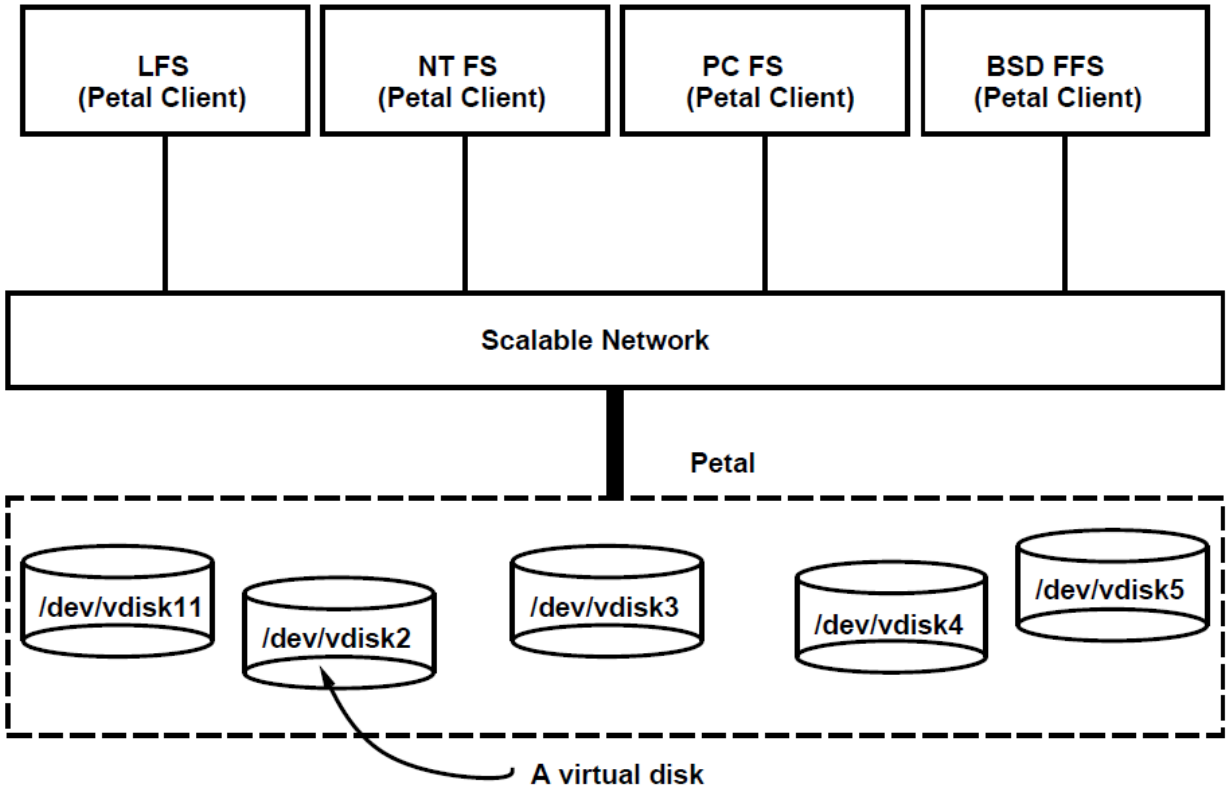
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

HY590.45
Modern Topics in Scalable Storage Systems

Kostas Magoutis
magoutis@csd.uoc.gr
http://www.csd.uoc.gr/~hy590-45

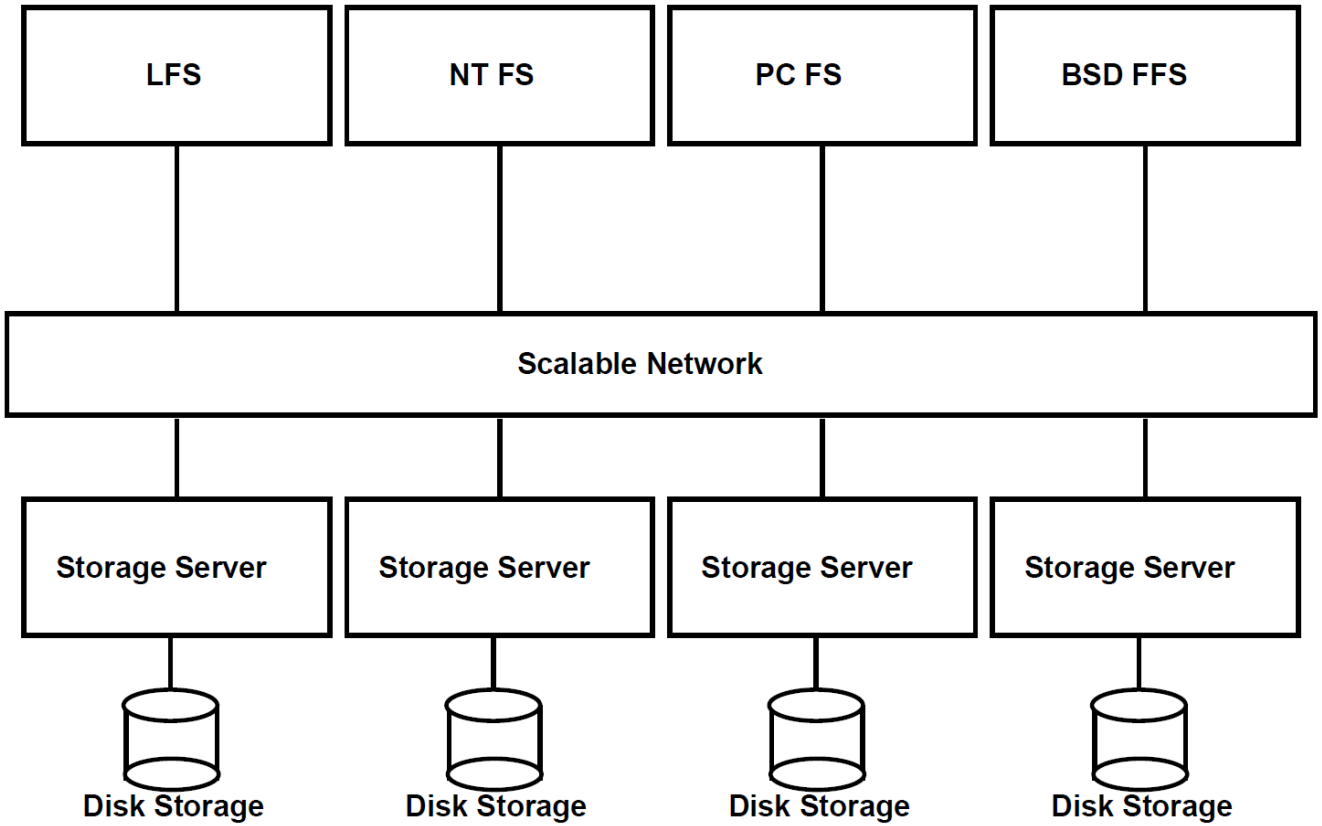
Petal: Client view



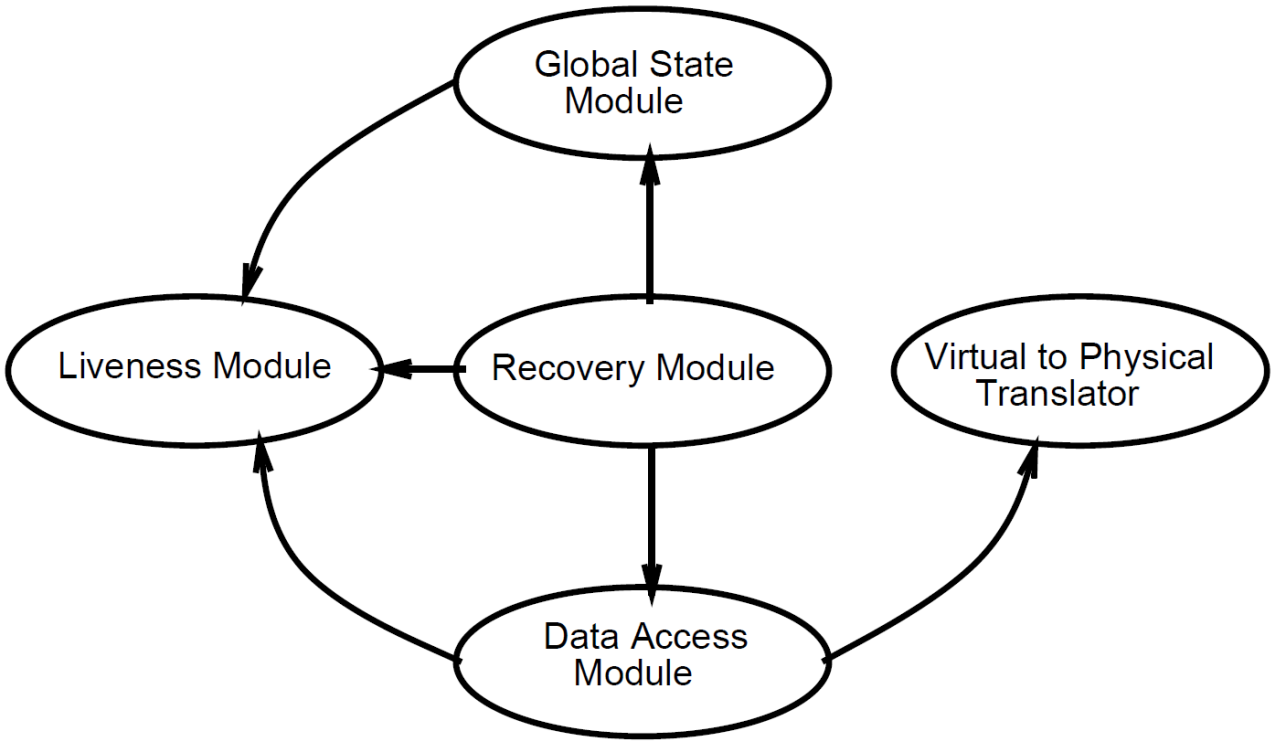
Goals of Petal

- Tolerate and recover from any component failure
- Geographically distribute to tolerate site failures
- Transparently reconfigure to expand, balance load
- Dynamically balance load
- Fast and efficient support for backups and recovery

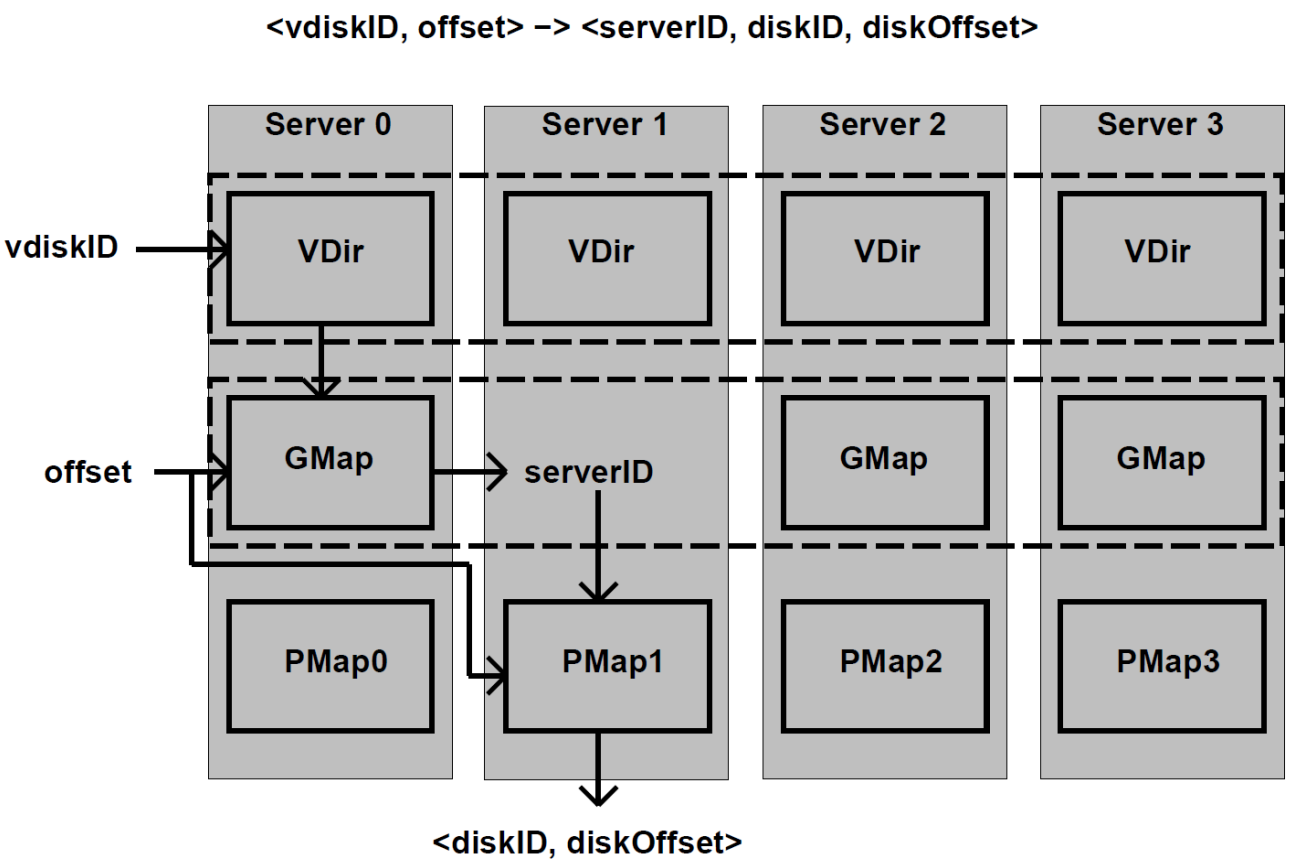
Physical structure



Server modules



Virtual to physical mapping



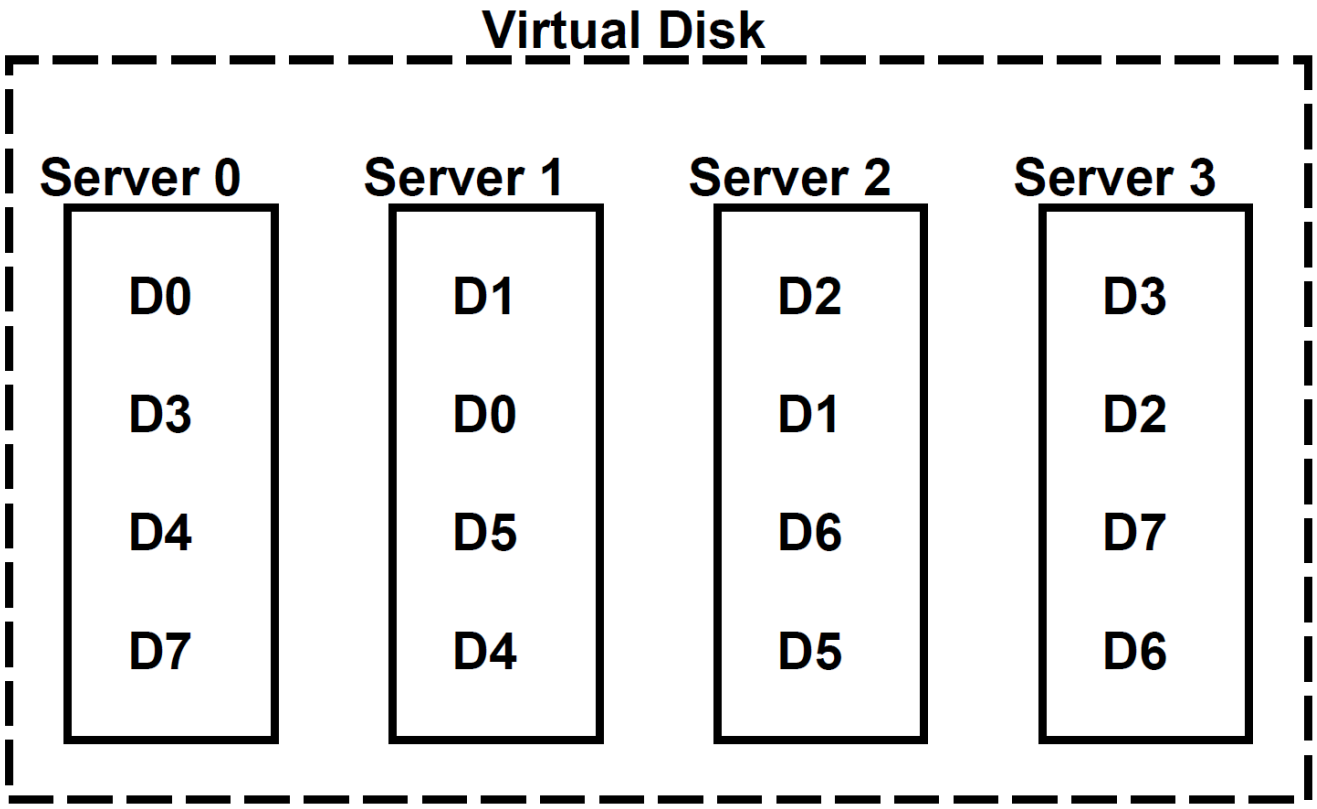
Support for backup

- Snapshots
  - Petal can quickly create an exact copy of a virtual disk at a specified point in time by using copy-on-write techniques
  - A snapshot is like any other vdisk, but cannot be modified
  - VDir: vdiskID → <global-map-identifier, epoch-number>
- Crash-consistent snapshot
  - Similar to disk image left after an application crash

Reconfiguration

- Dynamic change in vdisk # of servers, redundancy
- How is it performed
  - Create new GMap with desired redundancy, server mapping
  - Change VDir entries that refer to old GMap to new one
  - Redistribute data to the servers according to new GMap, requiring substantial amounts of network and disk traffic
  - Read requests will be tried on new GMap first, then the old GMap if the translation has not yet been transferred
  - Writes are always performed on the new GMap
- Improve efficiency via fencing

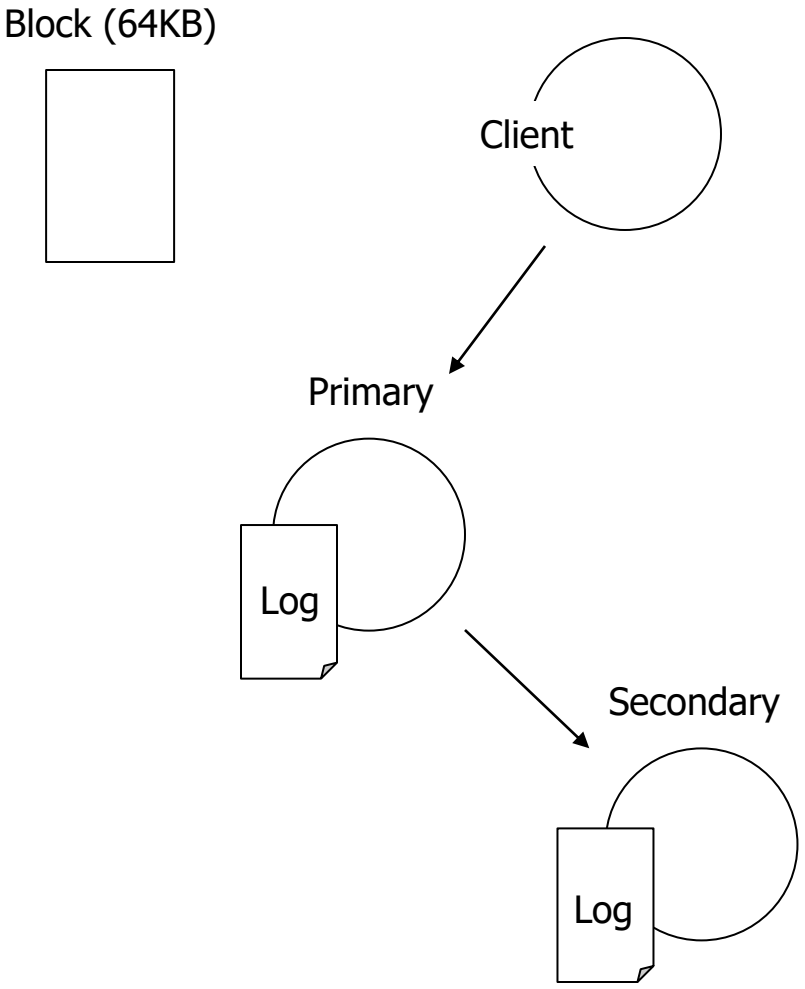
Chained de-clustering



Data access and recovery

- How are reads performed
- How are writes performed

Data access and recovery



Read protocol

1. Try primary; if live, get read lock
2. If down, try secondary; if live, get read lock
3. Return block contents, release read lock

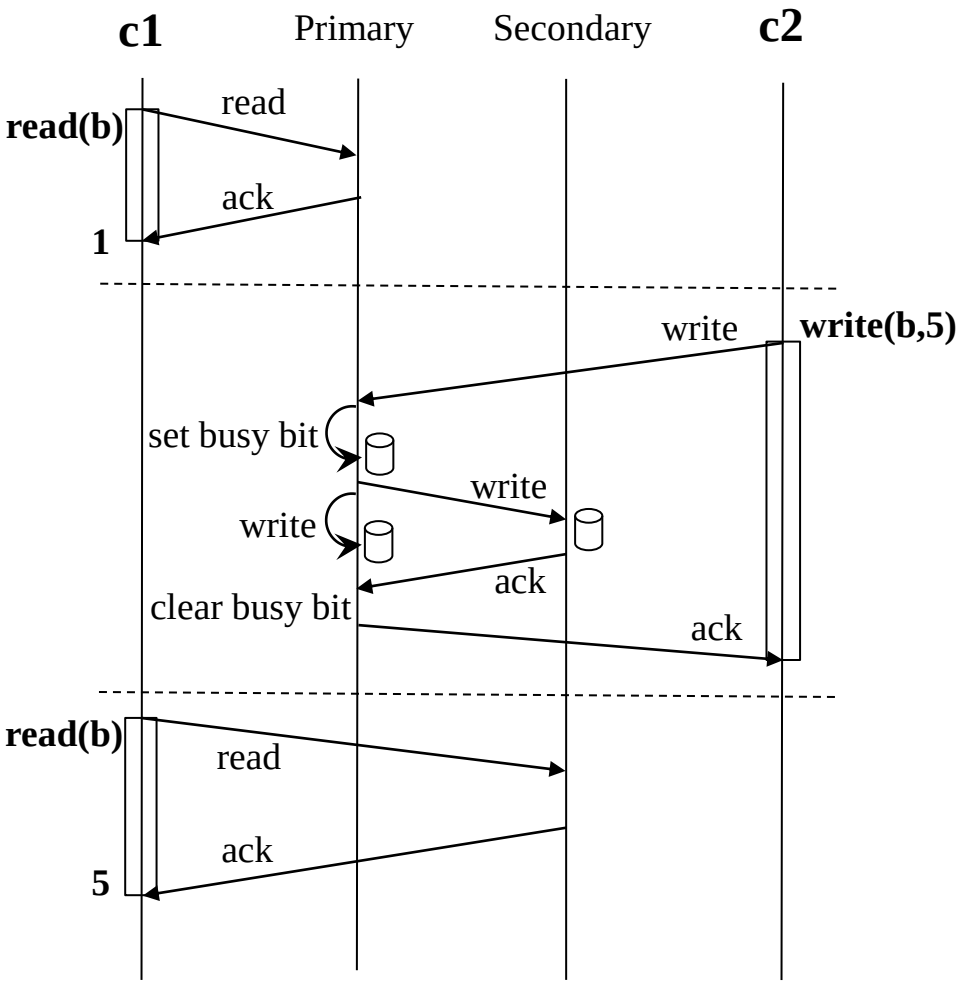
Write protocol

1. Contact primary
2. If alive, mark block busy in stable storage
3. Primary apply request locally and simultaneously send write to secondary
4. When both complete, clear busy bit, respond to client
5. If primary crashes during request, busy bit is used to recover later on
6. If primary dead, start from secondary
7. (Secondary checks primary indeed crashed)

If primary or secondary is down, on live node:

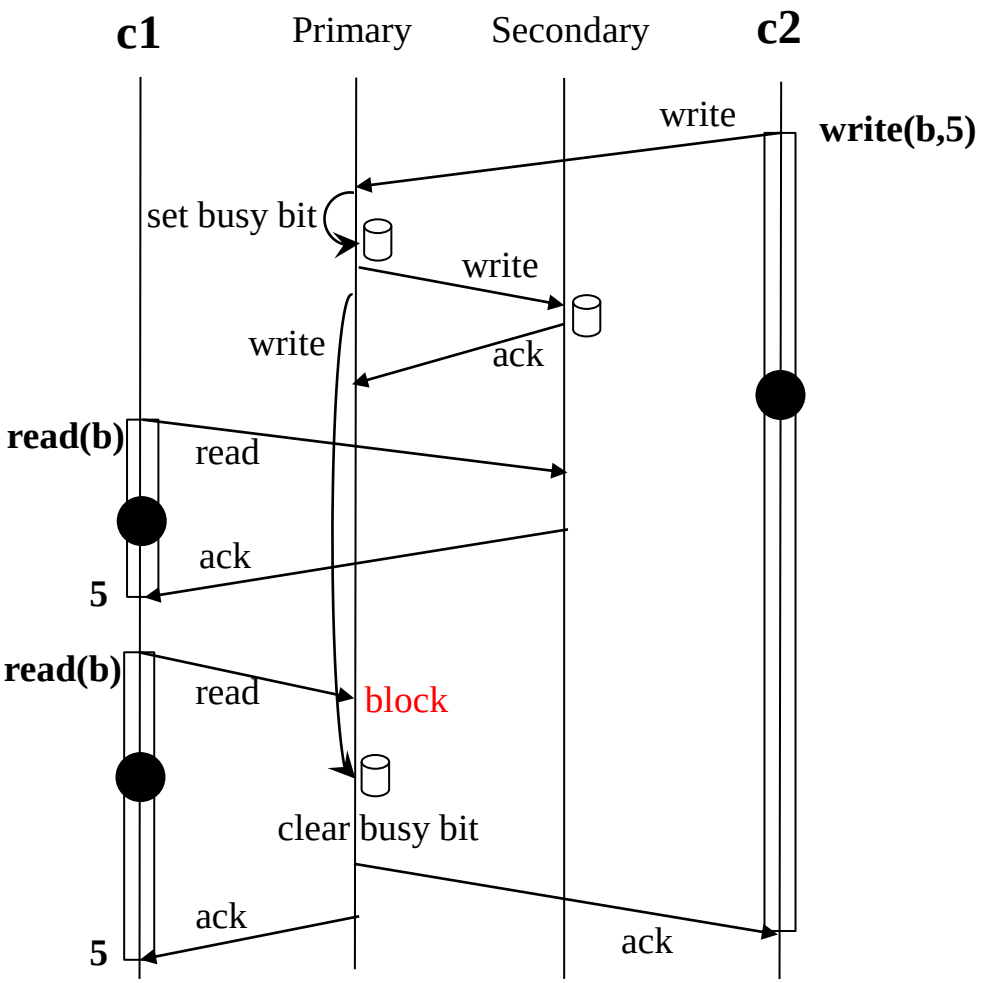
1. Mark data as *stale* before writing to disk
2. During recovery, make replicas consistent by exchanging *dirty-region log*

Failure-free operation



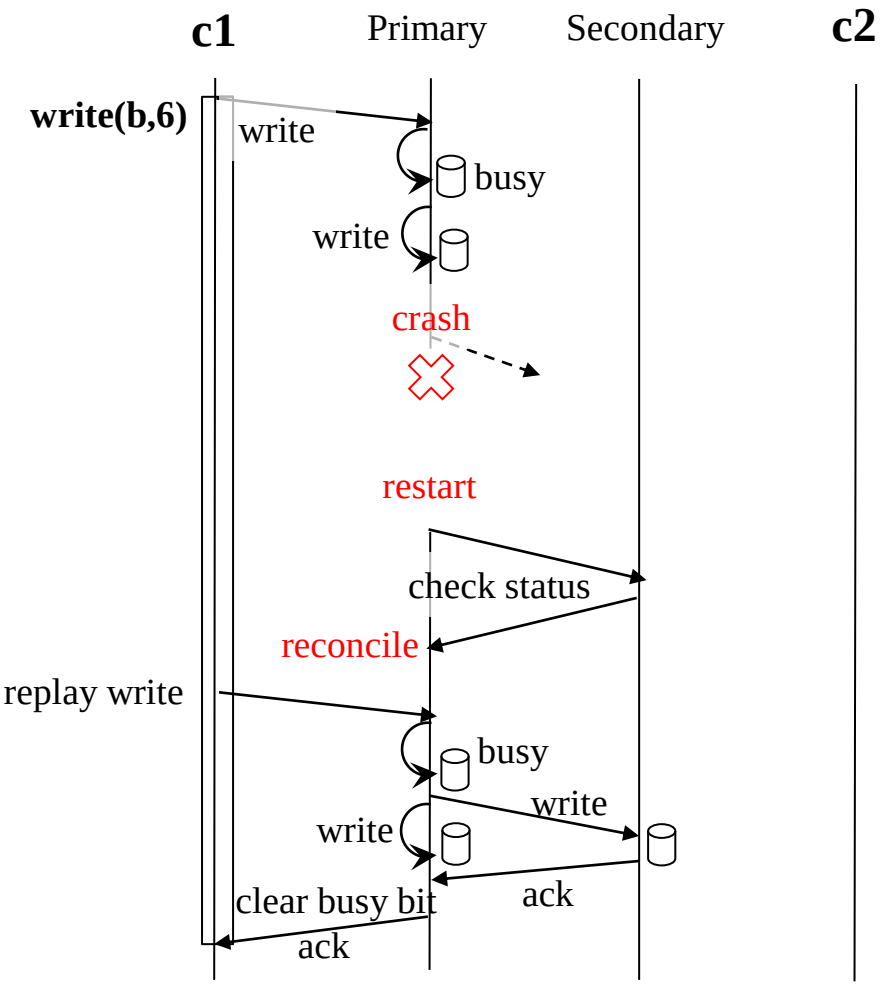
a read will see a state at least as recent as that produced by the most recently completed write that completed before the read started

Failure-free operation

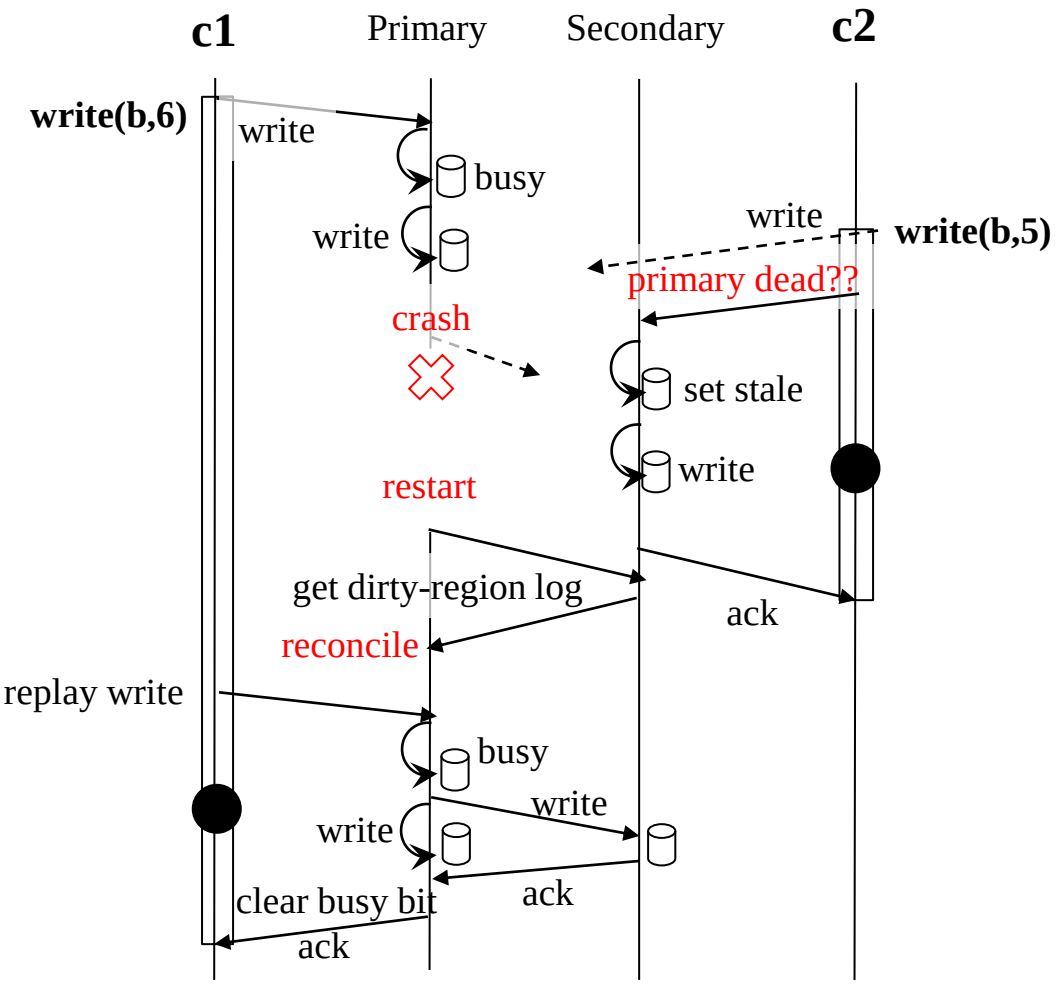


if some reader sees the results of a particular write, then any reader that starts after that reader finishes will also see a result at least that recent.

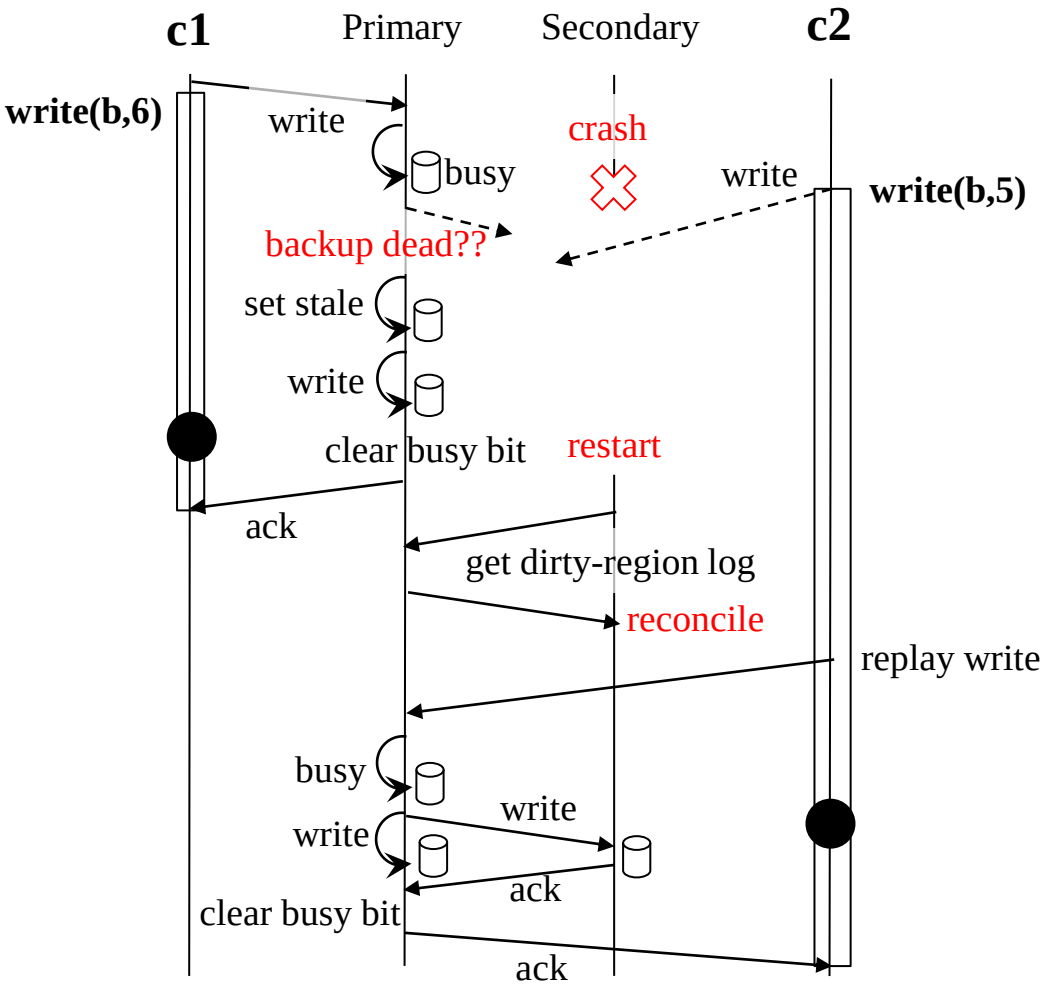
Recovery from primary crash



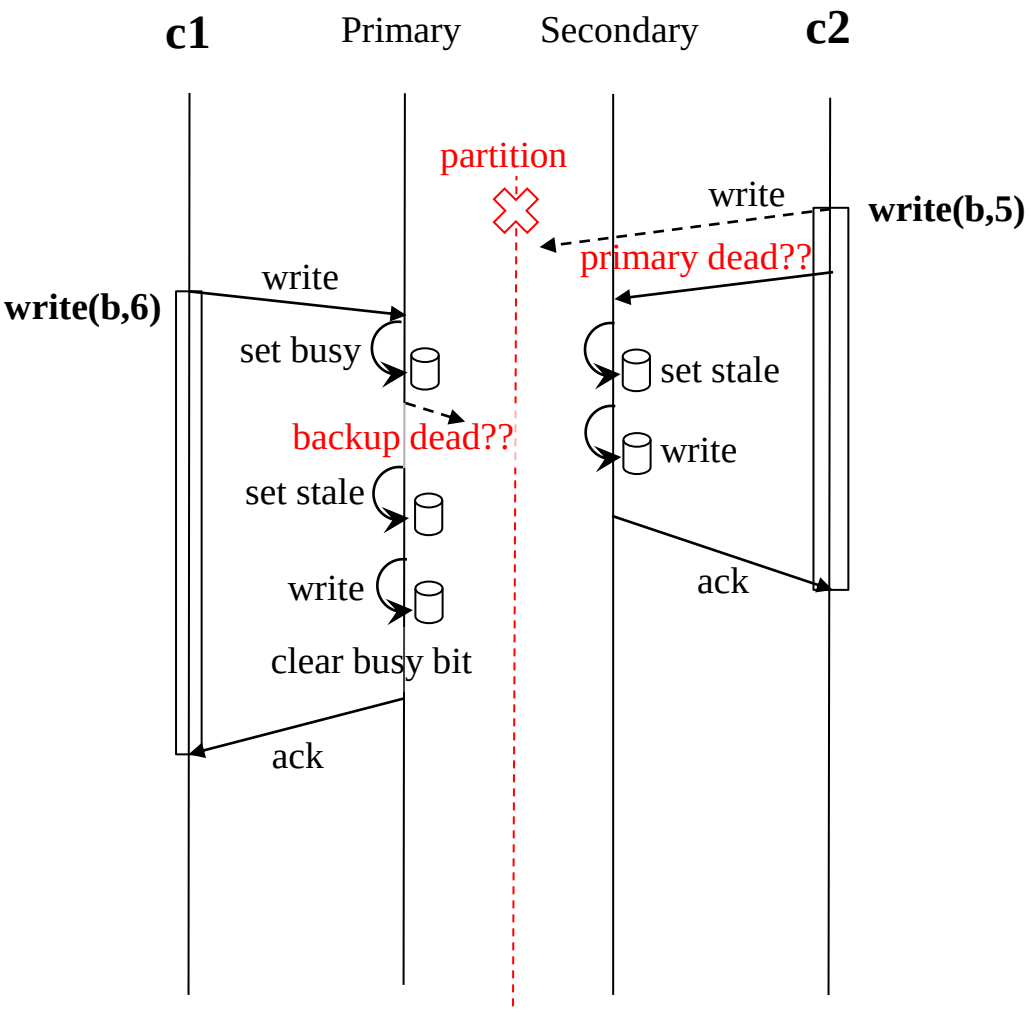
Recovery from primary crash



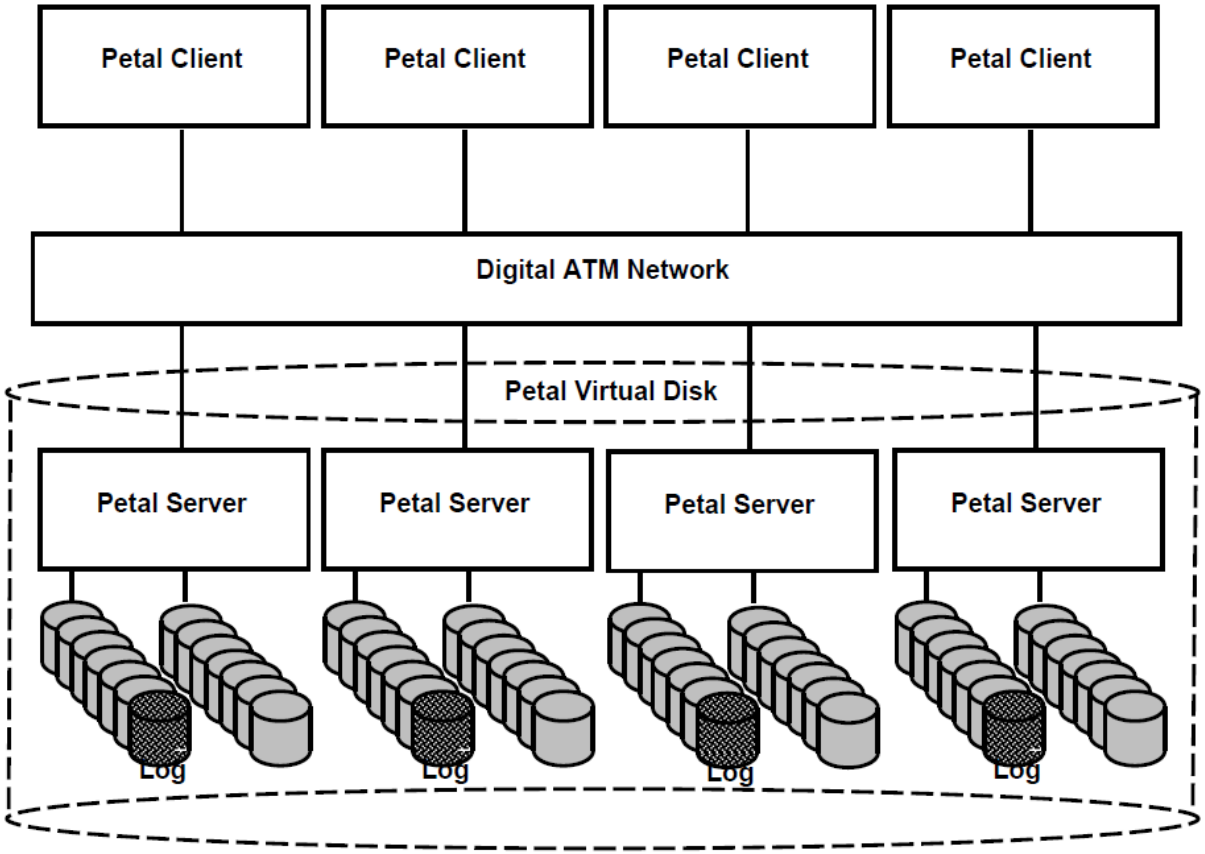
Recovery from secondary crash



Petal cannot handle partitions



Prototype


Petal Client
Petal Client
Petal Client
Petal Client
Digital ATM Network
Petal Virtual Disk
Petal Server
Petal Server
Petal Server
Petal Server
Log
Log
Log
Log

Latency, throughput results

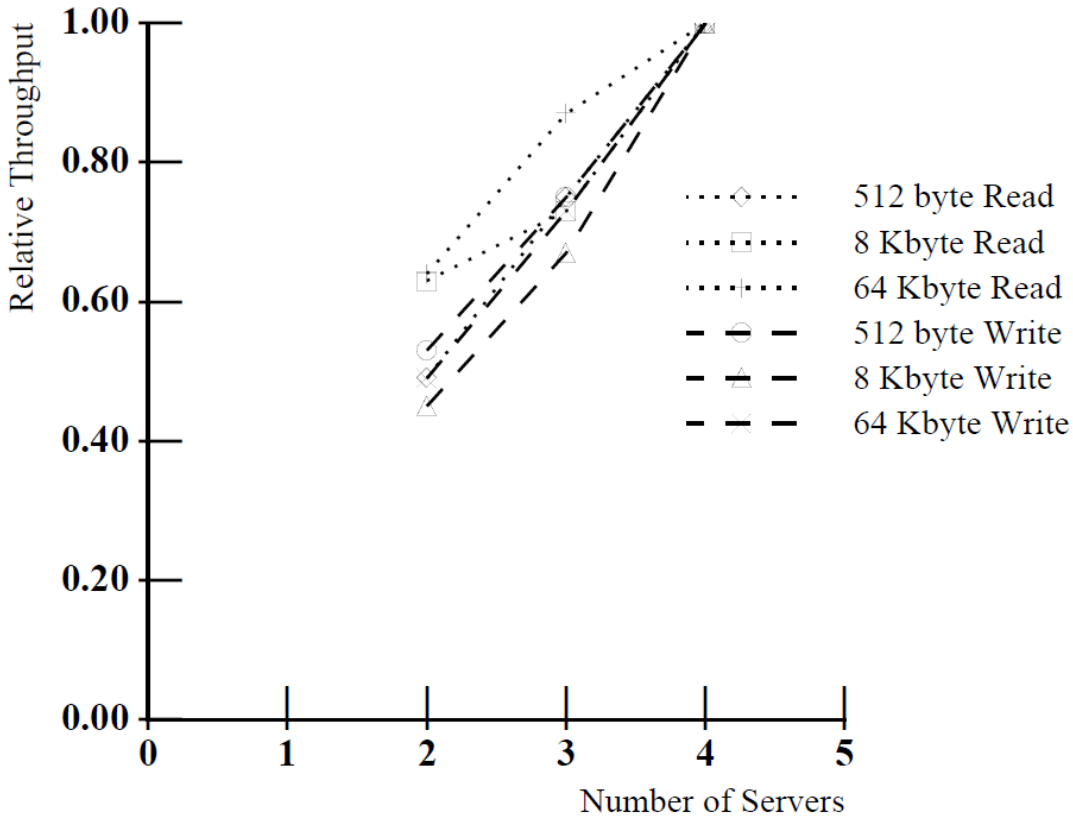
Client Request Latency (ms)			
Request	**Local Disk**	**Petal**	
		RZ29 Log	**NVRAM Log**
512 byte Read	9	10	10
8 Kbyte Read	11	12	12
64 Kbyte Read	21	28	28
512 byte Write	10	19	12
8 Kbyte Write	12	22	16
64 Kbyte Write	20	40	33

Table 1: Latency of a Chained-Declustered Virtual Disk

Aggregate Throughput (RZ29 Log)			
Request	**Normal**	**Failed**	**% of Normal**
512 byte Read	3150 req/s	2310 req/s	73 %
8 Kbyte Read	20 Mbytes/s	14.6 Mbytes/s	73 %
64 Kbyte Read	43.1 Mbytes/s	33.7 Mbytes/s	78 %
512 byte Write	1030 req/s	1055 req/s	102 %
8 Kbyte Write	6.6 Mbytes/s	6.6 Mbytes/s	100 %
64 Kbyte Write	12.3 Mbytes/s	12.5 Mbytes/s	101 %

Table 2: Normal and Failed Throughput of a Chained-Declustered Virtual Disk

Scaling


Relative Throughput
1.00
0.80
0.60
0.40
0.20
0.00
0
1
2
3
4
5
Number of Servers
512 byte Read
8 Kbyte Read
64 Kbyte Read
512 byte Write
8 Kbyte Write
64 Kbyte Write